The nonlinear matrix-Riccati equation has many solutions, but only <u>one</u> leads to a stable feedback.

Algorithm: (without proof)

① Check controllability. If not completely controllable, input-decouple the uncontrollable modes first.

② Compute the Hamiltonian matrix of the controllable subsystem:

$$H = \left[\begin{array}{c|c} (F + GR^{-1}G'(F')^{-1}Q) & -GR^{-1}G'(F')^{-1} \\ \hline -(F')^{-1}Q & (F')^{-1} \end{array}\right]$$

③ Compute the spectral decomposition of H:

$$[V, \Lambda] = Eig(H)$$

If the system is controllable, H will have exactly n eigenvalues inside the unit circle and n outside the unit circle, and <u>none</u> on the unit circle itself.

④ Take the subset of the right Modal matrix that relates to eigenvalues inside the unit circle:

$$\hat{V} = \begin{bmatrix} & n \\ & \\ & 2n \end{bmatrix}$$

⑤ Cut $\hat{V}$ into an upper and a lower portion:

$$\hat{V} = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} V_1 & n \\ \hline V_2 & n \end{bmatrix}$$

⑥ Compute the solution to the algebraic Riccati equation:

$$P = V_2 \cdot V_1^{-1}$$

(Notice: Any other combination of n eigenvectors would have given us another solution, but this is the one and only solution that leads to a stable closed-loop system.)

⑦ $K = (R + G'PG)^{-1} G'PF$

is the desired state feedback.

This is a numerically benign algorithm (numerically <u>much</u> better than pole placement).

⑧ $\underline{u} = \underline{r} - K \cdot \underline{x}$

In Matlab:

```
[> Ph22 = INV(F');
[> Ph12 = - G/R * G' * Ph22;
[> Ph11 = F - Ph12 * Q;
[> Ph21 = - Ph22 * Q;
[> H = [Ph11, Ph12; Ph21, Ph22];
[> [V,L] = EIG(H);
[>  indx = SORT(ABS(DIAG(L)));
[>  V1 = V(1:n, indx(1:n));
[>  V2 = V(n+1:2*n, indx(1:n));
[>  P = V2/V1;
[>  K = (R + G'*P*G)\G'*P*F;
```

This can easily be coded into a CTRL-C function:

```
[>  [K,P] = DLQR(F,G,Q,R)
```

↑

discrete linear quadratic (gaussian) regulator.

Disadvantages: DLQR has a tendency of placing the poles too close to each other ⟹ increases the sensitivity to parameter changes.

Possible Solution: There meanwhile exist techniques to individually move pole locations in the Riccati design of continuous systems. Nobody tried to extend this technique to discrete time systems, but this should be possible (open research ⟹ good topic for an MS Thesis).

- DLQR and PLACE can both be used for state-feedback design.
  ⟹ We can obtain output feedback by solving the DLQR problem once for the controller, and once for the observer:

```
[> k = DLQR(F, G, Q, R);
[> l = DLQR(F', H', Q0, R0)';
```

Output weighting:

Sometimes, it is desirable to weight the outputs instead of the states:

$$\left| \begin{array}{l} \underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot \underline{u}(k) \\ \underline{y}(k) = H \cdot \underline{x}(k) \end{array} \right| \quad (\underline{I = \emptyset})$$

$$PI = \sum_{i=0}^{\infty} \{ \underline{y}_i' Q \underline{y}_i + \underline{u}_i' R \underline{u}_i \} \stackrel{!}{=} \underset{\forall \underline{u}(t)}{Min}$$

$$Q \geq \emptyset \; ; \quad R > \emptyset$$

$$\underline{y}'(i) \cdot Q \cdot \underline{y}(i) = (H \underline{x}(i))' Q \cdot (H \underline{x}(i))$$

$$= \underline{x}'(i) \cdot \underbrace{H' Q H}_{Q_n} \cdot \underline{x}(i) \qquad Q_n \geq \emptyset$$

$$\Leftrightarrow PI = \sum_{i=\emptyset}^{\infty} \{ \underline{x}_i' Q_n \underline{x}_i + \underline{u}_i' R \underline{u}_i \} \stackrel{!}{=} \underset{\forall \underline{u}(t)}{Min}$$

is the same problem.

If $\underline{I \neq \emptyset}$, the same concept can also be used, but it

is a little more tricky:

$$\underline{y}'(i)\,Q\,\underline{y}(i) = \left(H\underline{x}(i) + I\underline{u}(i)\right)' Q \left(H\underline{x}(i) + I\underline{u}(i)\right)$$

$$= \left(\underline{x}'(i)H' + \underline{u}'(i)I'\right) Q \left(H\underline{x}(i) + I\underline{u}(i)\right)$$

$$= \underline{x}'(i)\cdot H'QH\cdot\underline{x}(i) + \underline{x}'(i)\cdot H'QI\cdot\underline{u}(i)$$
$$+ \underline{u}'(i)\cdot I'QH\cdot\underline{x}(i) + \underline{u}'(i)\cdot I'QI\cdot\underline{u}(i)$$

$$= \underline{x}'(i)\cdot\hat{Q}\cdot\underline{x}(i) + \underline{x}'(i)\cdot\hat{N}\cdot\underline{u}(i) + \underline{u}'(i)\cdot\hat{N}'\cdot\underline{x}(i)$$
$$+ \underline{u}'(i)\cdot\tilde{R}\cdot\underline{u}(i)$$

where:
$$\hat{Q} = H'QH$$
$$\tilde{R} = I'QI$$
$$\hat{N} = H'QI$$

or:

$$PI = \sum_{i=\phi}^{\infty}\left\{\underline{x}_i'\hat{Q}\,\underline{x}_i + \underline{x}_i'\hat{N}\underline{u}_i + \underline{u}_i'\hat{N}'\underline{x}_i + \underline{u}_i'\hat{R}\underline{u}_i\right\} \doteq \min_{\forall\,\underline{u}(t)}$$

where: $\hat{R} = R + \tilde{R} = R + I'QI$

or:

$$PI = \sum_{i=\phi}^{\infty}\left[\underline{x}_i'\ \underline{u}_i'\right]\cdot\begin{bmatrix}\hat{Q} & \hat{N}\\ \hat{N}' & \hat{R}\end{bmatrix}\cdot\begin{bmatrix}\underline{x}_i\\ \underline{u}_i\end{bmatrix} \doteq \min_{\forall\,\underline{u}(t)}$$

Another variable transformation can remove the mixed terms:

$$PI = \sum_{i=0}^{\infty} \{ \underline{x}_i' (\hat{Q} - \hat{N}\hat{R}^{-1}\hat{N}') \underline{x}_i + (\underline{u}_i + \hat{R}^{-1}\hat{N}'\underline{x}_i)' \cdot \hat{R} \cdot (\underline{u}_i + \hat{R}^{-1}\hat{N}'\underline{x}_i) \} \overset{!}{=} \min_{\forall \underline{u}(t)}$$

as can be easily verified by multiplying out.

$\Rightarrow$ Let:

$$\boxed{\begin{aligned} Q_n &= \hat{Q} - \hat{N}\hat{R}^{-1}\hat{N}' \\ \underline{u}_n &= \underline{u} + \hat{R}^{-1}\hat{N}'\underline{x} \end{aligned}}$$

$$\Rightarrow PI = \sum_{i=0}^{\infty} \{ \underline{x}_i' Q_n \underline{x}_i + \underline{u}_{ni}' \hat{R} \underline{u}_{ni} \} \overset{!}{=} \min_{\forall \underline{u}_n(t)}$$

is a performance index with state weighting and without mixed terms for a modified problem:

$$\boxed{\begin{aligned} \underline{x}(k+1) &= F_n \underline{x}(k) + G_n \underline{u}_n(k) \\ \underline{y}(k) &= H_n \underline{x}(k) + I_n \underline{u}_n(k) \end{aligned}}$$

$$\underline{y}(k) = H\underline{x}(k) + I\underline{u}(k)$$
$$= H\underline{x}(k) + I\underline{u}_n(k) - I\hat{R}^{-1}\hat{N}'\underline{x}(k)$$

$$\Rightarrow \underline{y}(k) = \underbrace{\left[H - I\hat{R}^{-1}\hat{N}'\right]}_{H_n}\underline{x}(k) + \underbrace{I}_{I_n}\cdot\underline{u}_n(k)$$

$$\underline{x}(k+1) = F\underline{x}(k) + G\underline{u}(k)$$
$$= F\underline{x}(k) + G\underline{u}_n(k) - G\hat{R}^{-1}\hat{N}'\underline{x}(k)$$

$$\Rightarrow \underline{x}(k+1) = \underbrace{\left[F - G\hat{R}^{-1}\hat{N}'\right]}_{F_n}\underline{x}(k) + \underbrace{G}_{G_n}\underline{u}_n(k)$$

$\Rightarrow$ The output weighting problem can be reformulated as follows:

$$\boxed{\begin{aligned}\underline{x}(k+1) &= F_n\cdot\underline{x}(k) + G\underline{u}_n(k)\\ \underline{y}(k) &= H_n\underline{x}(k) + I\underline{u}_n(k)\end{aligned}}$$

where:

$$\boxed{\begin{aligned}F_n &= F - G\hat{R}^{-1}\hat{N}'\\ H_n &= H - I\hat{R}^{-1}\hat{N}'\end{aligned}}$$

$$PI = \sum_{i=0}^{\infty}\left\{\underline{x}_i'Q_n\underline{x}_i + \underline{u}_n'\hat{R}\underline{u}_n\right\} \doteq \min_{\forall\,\underline{u}_n(+)}$$

with:

$$\begin{aligned}\hat{Q} &= H'QH \\ \hat{N} &= H'QI \\ \hat{R} &= R + I'QI \\ Q_n &= \hat{Q} - \hat{N}\hat{R}^{-1}\hat{N}'\end{aligned}$$

$$\rightarrow \hat{K} = \mathrm{DLQR}(F_n, G, Q_n, \hat{R})$$

delivers a state-feedback from $\underline{x}$ into $\underline{u}_n$:

$$\underline{u}_n = \underline{r} - \hat{K} * \underline{x} \equiv \underline{u} + \hat{R}^{-1}\hat{N}'\underline{x}$$

$$\Rightarrow \underline{u} = \underline{r} - \underbrace{\left[\hat{K} + \hat{R}^{-1}\hat{N}'\right]}_{K}\underline{x}$$

$\Rightarrow$ backtransformation:

$$K = \hat{K} + \hat{R}^{-1}\hat{N}'$$

gives the state-feedback from $\underline{x}$ into $\underline{u}$.

In Matlab :

Let us solve the problem :

$$\left| \begin{array}{l} \underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot \underline{u}(k) \\ \underline{y}(k) = H \cdot \underline{x}(k) + I \cdot \underline{u}(k) \end{array} \right|$$

$$PI = \sum_{i=0}^{\infty} [\underline{x}_i' \; \underline{u}_i'] \cdot \begin{bmatrix} \hat{Q} & \hat{N} \\ \hat{N}' & \hat{R} \end{bmatrix} \cdot \begin{bmatrix} \underline{x}_i \\ \underline{u}_i \end{bmatrix} \stackrel{!}{=} \underset{\forall \underline{u}(t)}{\text{Min}}$$

Algorithm :

```
[> FN = F - (G/R̂)*N̂';
[> QN = Q̂ - (N̂/R̂)*N̂';
[> K̂ = DLQR(FN, G, QN, R̂);
[> K = K̂ + (R̂\N̂');
```

This could be coded as :

```
[> K = DLQRN(F, G, Q̂, R̂, N̂);
```

Let us now solve the complete output weighting problem:

$$\left| \begin{array}{l} \underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot \underline{u}(k) \\ \underline{y}(k) = H \cdot \underline{x}(k) + I \cdot \underline{u}(k) \end{array} \right|$$

with:

$$PI = \sum_{i=\phi}^{\infty} [\underline{y}_i' \; \underline{u}_i'] \cdot \begin{bmatrix} Q & \phi \\ \phi & R \end{bmatrix} \cdot \begin{bmatrix} \underline{y}_i \\ \underline{u}_i \end{bmatrix} \stackrel{!}{=} \underset{\forall \underline{u}(t)}{\text{Min}}$$

Algorithm:

```
[> Q̂ = H'*Q*H;
[> R̂ = R + I'*Q*I;
[> N̂ = H'*Q*I;
[> K = DLQRN(F,G,Q̂,R̂,N̂);
```

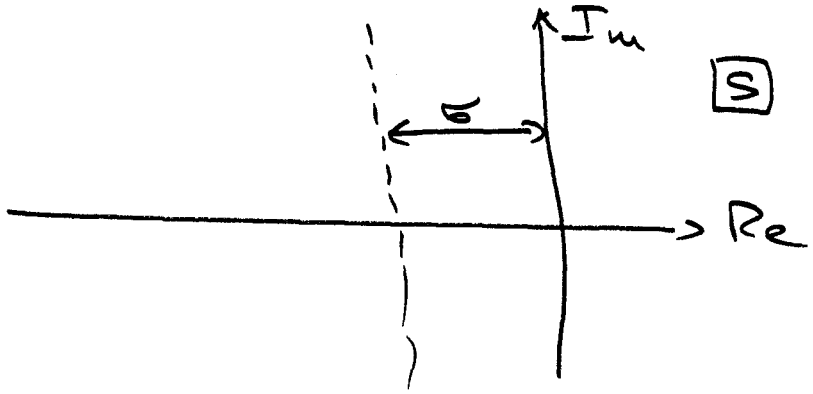
which can be made available as function:

```
[> K = DLQRY(F,G,H,I,Q,R);
```

Exponential Stability:

In order to guarantee a maximum settling time, we want all eigenvalues at least a factor of α inside the unity circle.

$$\sigma = \frac{4}{T_s}$$



$$\lambda_c = -\sigma \pm j\omega$$

$$\Rightarrow \lambda_d = \exp(\lambda_c T) = \exp(-\sigma \cdot T \pm j\omega T)$$

$$= \underbrace{\exp(-\sigma T)}_{\alpha} \cdot \underbrace{\exp(\pm j\omega T)}_{\{\text{unity circle}\}}$$

Given:

$$\left| \begin{array}{l} \underline{x}(k+1) = F \cdot \underline{x}(k) + G \cdot \underline{u}(k) \\ \underline{y}(k) = H \cdot \underline{x}(k) + I \cdot \underline{u}(k) \end{array} \right|$$

We can look at a different system:

$$\left| \begin{array}{l} \underline{x}_n(k+1) = \tilde{F} \cdot \underline{x}_n(k) + \tilde{G} \cdot \underline{u}(k) \\ \underline{y}_n(k) = H \cdot \underline{x}_n(k) + I \cdot \underline{u}(k) \end{array} \right|$$

where:

$$\left| \begin{array}{l} \tilde{F} = \frac{1}{\alpha} \cdot F \\ \tilde{G} = \frac{1}{\alpha} \cdot G \end{array} \right|$$

$$\Rightarrow Eig\{\tilde{F}\} \equiv Eig\{\tfrac{1}{\alpha}\cdot F\} \equiv \tfrac{1}{\alpha}\cdot Eig\{F\}$$

We now design the state-feedback with DLQR or one of its variants

$$\Rightarrow \tilde{F}_{CL} = \tilde{F} - \tilde{G}K$$

is stable.

- We now apply the <u>same</u> feedback to our original system:

$$\Rightarrow F_{CL} = F - G\cdot K$$

$$\Rightarrow F_{CL} = \alpha\cdot\tilde{F} - \alpha\cdot\tilde{G}\cdot K$$

$$= \alpha(\tilde{F} - \tilde{G}K) = \alpha\cdot\tilde{F}_{CL}$$

$$\Rightarrow \underbrace{Eig\{F_{CL}\}}_{\text{inside circle with radius } \alpha<1} = \alpha\cdot \underbrace{Eig\{\tilde{F}_{CL}\}}_{\text{stable}}$$

as desired.